AIAA 2001-4373

Building Parametric and Probabilistic Dynamic Vehicle Models Using Neural Networks

Julien Scharl *
Dimitri Mavris †
Aerospace Systems Design Laboratory,
Georgia Institute of Technology
Atlanta, GA 30332–0150

Abstract

During the past decade, the aircraft vehicle design process has undergone a major shift of focus from pure performance towards a balance between vehicle characteristics and cost, namely affordability. In addition, accelerated advances in computing technology have helped render a complete parametric and probabilistic design process feasible. All of these changes have allowed more knowledge to be brought earlier into the design process, which helps designers make more informed and therefore better decisions, earlier in the design process. Computing power now allows extensive physics-based vehicle modeling early in the design cycle. A full non-linear six degree of freedom parametric dynamic vehicle model should be attainable as early as the conceptual design phase. Such a vehicle model would help understand the effects of design variables on vehicle characteristics and operation through analysis and simulation. Furthermore, probabilistic design methods allow for the proper treatment of uncertainty and fidelity inherent in such a model. This paper formulates a framework to arrive at a conceptual non-linear six degree of freedom parametric and probabilistic dynamic vehicle model based on neural networks.

Introduction

IN response to rapid changes in the world economic and socio-political environment, the aerospace system design process is being overhauled. The new paradigm in aerospace design calls for shorter design cycles with strong focus on affordability and quality. As a result, design methodologies being developed today revolve around parametric and probabilistic design environments with strong focus on product life cycle cost.[1–4] Furthermore, rapid advances in computing technology may now make a complete *virtual*, parametric and probabilistic design process possible. Such a design process is widely believed to be the optimal means of achieving this paradigm shift.[4–7]

Implementations of such methodologies at the conceptual design phase currently rely on simplified disciplinary models. Traditional disciplines such as aerodynamics, structures and propulsion are well represented in early phases of the design process and all participate in the synthesis and sizing process, the central activity during conceptual design. This results in vehicles appropriately sized for a given mission yet prevents early design decisions related to empennage configurations and control surface sizes with respect to stability, control and handling requirements. This limitation is due to the oversimplification of the problem resulting from a shortage of design knowledge and the presence of uncertainty. This can ultimately result in designs which fail to meet customer or certification requirements. In those cases, needed design modifications resulting from flight testing and sophisticated piloted simulations in late design stages can easily generate insurmountable development costs.

Problem statement

In order to capture the needed metrics for empennage and control surface sizing, it becomes necessary to mathematically express vehicle dynamics in terms of the design variables.

Traditional vehicle dynamic and control analyses are based on static stability and control derivatives and linear models, without actually arriving at a full dynamic model. DeLaurentis developed parametric and probabilistic models for handling qualities evaluation in conceptual design.[8,9] However, these models are limited due to their inherent simplicity resulting from assumptions made about the vehicle motion. These limitations can be lifted with the creation of full dynamic vehicle models, which are animated via differential equation solvers; this is the basis of flight simulation.

A design methodology incorporating flight simu-

*Graduate Research Assistant, AIAA student member

†Professor, Dir. ASDL, Boeing Chair in Advanced Aerospace Systems Analysis, Associate Fellow AIAA

lation has been proposed and tested by the authors.[10] The Virtual Testing and Evaluation Methodology (VT&E) is centered around an automatic input data generator specific to a flight simulation program (VATES),[11] which is used to simulate the vehicle in various flight regimes. However, the methodology is limited to vehicle models tailored for VATES (i.e. standard stability and control derivative tables). Its formulation lacks the capability for parametric evaluations of simulation output as functions of design variables.

To address the need for appropriate tail and control surface sizing in conceptual design, the authors propose a full dynamic vehicle model which is parametric in the flight conditions as well as the design variables.

Models and metamodels in design

Central to the aerospace design process is the ability to model the system in question. Modeling takes place at all levels in all design disciplines. Models vary from a simple equation to an intricate piece of computer code. Physics-based models are preferable and become necessary for exotic and revolutionary concepts that lie outside the range of validity of regression-based models. Examples of physics-based models include potential flow aerodynamic models and finite element models. In many instances, implementing these models within a parametric design environment can slow down the design process tremendously due to long execution times. This can be remedied using models of these models, or *metamodels*. In essence, the metamodel is a faster and more efficient replacement for the full model in design analyses. It is obtained by mapping the model outputs to its inputs.

Metamodeling techniques

Recent advances in design methodology have made extensive uses of Design of Experiments (DOE) and Response Surface Methodology (RSM)[12] to generate disciplinary metamodels, which are used during the sizing and synthesis process.[3,13] Response surfaces are usually limited to the second degree due to the large amount of terms that arise from increasing the order for a large number of inputs. Due to this limitation, it becomes questionable to use Response Surfaces to fit a model whose characteristics are either unknown, of a high degree, or contains a large set of input variables.

Other less limiting metamodeling techniques need to be investigated. Weiss has shown that local model networks can successfully be applied to the aerodynamic model identification problem.[14] Several authors have compared Neural Networks to polynomial-based metamodeling techniques.[15–17] Neural Networks were chosen for this study because of their universal function approximation capability and their ease of implementation within the *Matlab/Simulink* environment.

Dynamic vehicle models

A dynamic vehicle model seeks to represent vehicle motion in terms of differential equations. This is accomplished by developing equations of motion for the vehicle. For a six degree of freedom model, the equations of motion of a rigid body (of constant mass) are:

$$\left.\begin{array}{l} m\left(\dot{u}+qw-rv\right)=F_x \\ m\left(\dot{v}+ru-pw\right)=F_y \\ m\left(\dot{w}+pv-qu\right)=F_z \end{array}\right\} \tag{1}$$

$$\left.\begin{array}{l} I_x\dot{p}-I_{yz}(q^2-r^2)-I_{zx}(\dot{r}+pq) \\ -I_{xy}(\dot{q}-rp)-(I_y-I_z)qr=L \\ \\ I_y\dot{q}-I_{zx}(r^2-p^2)-I_{xy}(\dot{p}+qr) \\ -I_{yz}(\dot{r}-pq)-(I_z-I_x)rp=M \\ \\ I_z\dot{r}-I_{xy}(p^2-q^2)-I_{yz}(\dot{q}+rp) \\ -I_{zx}(\dot{p}-qr)-(I_x-I_y)pq=N \end{array}\right\} \tag{2}$$

Where u, v, w are ground velocities in the body x, y, and z direction respectively; p, q, r are rotational velocities with respect to the inertial frame around the x, y, and z body axes respectively; F_x, F_y, F_z, L, M, N are the external forces and moments applied on the body in the body x, y and z directions respectively.[18–20]

The goal is to express the forces and moments as functions of both state (flight conditions) and design variables. For conceptual design studies, we will assume that all forces on the body originate from its aerodynamics, propulsion and gravity only. Schematically, the process of arriving at parametric dynamic models is shown in Figure 1.

Using metamodels, the aerodynamic and propulsive forces and moments, as well as the vehicle's mass properties, are expressed as functions of operational *and* design variables. The authors believe that an aerodynamic model based on total aerodynamic coefficients is superior to a more traditional flight mechanics model based on stability derivatives. The dynamic model only requires total forces and moments acting on the body. Splitting these (or their respective coefficients) into a linear Taylor series approximation in terms of stability derivatives introduces error into the model since one will have to reconstruct the total forces and moments from the Taylor model.

Two degree of freedom (DOF) sizing is retained as the means of obtaining vehicle mass properties and engine characteristics. The process guarantees a vehicle of appropriate mass and engine through lift/weight and thrust/drag matching. A point mass model is sufficient to obtain such quantities through the generic design mission since no detailed flight maneuvers are needed for this process.

The outcome is a dynamic model of the vehicle that is parametric in both the design variables and flight conditions. In essence, it is a dynamic model

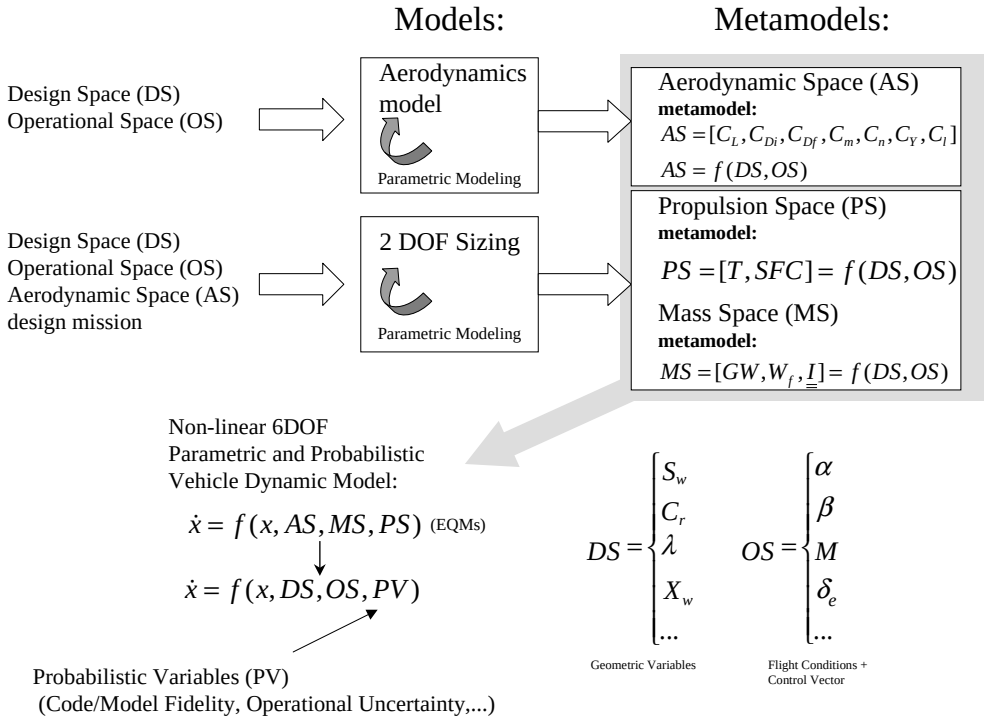


Fig. 1 Generating the parametric dynamic vehicle model

of the entire design space. Furthermore, by introducing random variables to model the fidelity of each metamodel and its corresponding model, the overall dynamic model becomes probabilistic. It can be used to analyze vehicle dynamic characteristics such as open loop handling qualities in terms of the frequency and damping of phugoid, short period and dutch roll modes of motion. The model can also be used in control analysis and design. The main benefit of such a model lies in the added vision of the effect of the design variables as well as model and metamodel fidelity on the dynamic characteristics of the vehicle. This will enable a designer to make appropriate sizing decisions with respect to the vehicle's operation.

Approach

This paper applies the process shown in Figure 1 to generate the parametric dynamic model for a generic 150 passenger subsonic transport. The baseline for this concept is shown in Figure 2. Neural Networks are used to generate the aerodynamic, propulsion and mass properties metamodels. These networks are then incorporated in the dynamic model mathematically expressed by equations 1 and 2 in *Matlab/Simulink*. This paper focuses on the creation of the metamodels used by the dynamic model and issues related to the use of neural networks for such metamodels. The dynamic model itself is not used in this study.

Aerodynamic modeling

The model chosen for the aerodynamic forces and moments is HASC, a potential flow computer model with a semi-empirical vortex lift model.[21] HASC

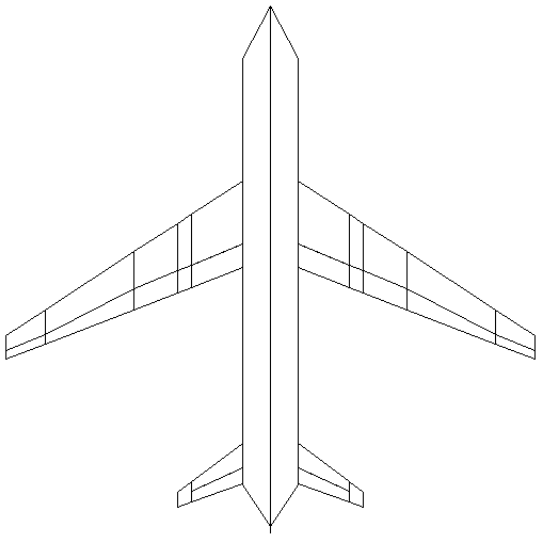


Fig. 2 Top view (HASC representation) of 150 passenger subsonic transport baseline

outputs the six force and moment aerodynamic coefficients in either body, stability or wind axes with changes in Mach number, angle of attack (α), angle of sideslip (β) and rotational rates (p, q and r). An object-oriented preprocessor to HASC was developed to automatically generate input files for a given geometry at a given operating condition (including control surface deflections). This preprocessor was used to generate cases varying 14 variables for the longitudinal data and 16 variables for the lateral data within a defined design space (shown in Table 1). A total of 3500 cases were obtained to create the metamodel.

This study did not make use of DOEs to generate

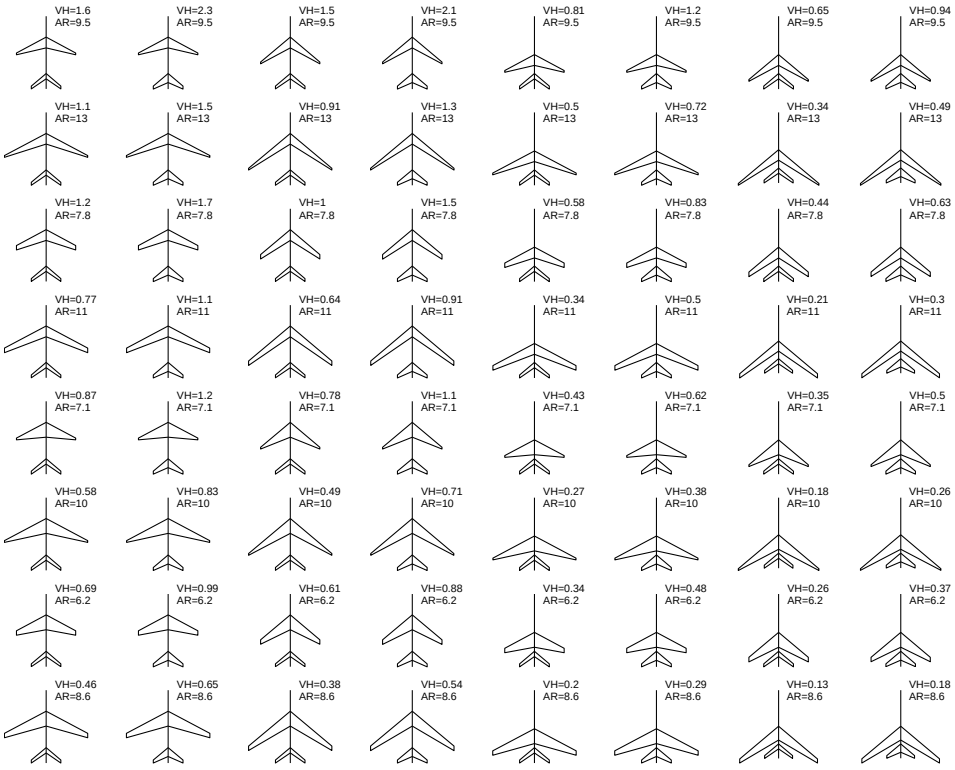


Fig. 3 Sample design space representation

model inputs. The ranges of the design variables are intentionally kept large to allow for the metamodels to be applicable through a large portion of the design space. DOEs typically select points at the corners of the design space. The authors found that a lot of these corners correspond to configurations for which HASC has difficulty computing data. A DOE of 14 variables targeting main effects, interactions and quadratic terms has 256 cases. On average, 25% of these cases had no valid output from the HASC program. By comparison, when a fixed number of random cases were generated, only 6 to 10 % of the desired number of cases had to be discarded. Lastly, the amount of time needed to generate the input data for the metamodels was far less than the time needed to optimize and train all neural networks. Time savings obtainable through a DOE approach would be insignificant compared to the overall time needed to generate the models.

Since HASC is based on potential flow theory, it is incapable of modeling friction drag. For this reason, a separate model was chosen for the friction drag coefficient, the Boeing Drag Analysis Program (BDAP).[22] The object-oriented preprocessor was extended to create BDAP input files. Again, 3500 random cases varying 15 variables tabulated in Table 1 were obtained to create the aerodynamic metamodel. Figure 3 shows a sample representation of 64 vehicle configurations chosen randomly from the design space. The figure also shows the horizontal tail volume coefficient $V_h = \frac{S_h l_h}{S\bar{c}}$ and wing aspect ratio for each of these cases to indicate the large variety of configurations captured by the metamodel.

Propulsion and mass/inertia modeling

Engine data for this study are obtained through the sizing and synthesis process via FLOPS.[23] Given a design mission specified in terms of segments (takeoff, climb, cruise etc.), FLOPS seeks to match lift with weight and thrust with drag throughout each segment using a 2 DOF model by iterating on the amount of fuel carried on board. The process starts by computing empty weight based on the given geometry using empirical relations and an initial guess for the gross weight. Also, given engine design data, a parametric engine deck is generated. The output of the process is a vehicle of appropriate weight and an engine of appropriate thrust for the given design mission.

For this study, engine design parameters were fixed and therefore one engine deck was generated for all vehicles in the design space. The engine model output by FLOPS is represented as tabular data for thrust and fuel consumption as functions of Mach number, altitude and throttle setting. This data is used for the propulsive force metamodel.

The vehicle's mass properties are direct outputs of FLOPS. Empty weight (W_e) and maximum fuel weight (W_f) are computed for a given configuration, design mission and payload capacity. The inertia matrix and nominal center of gravity are also computed by FLOPS given an estimate of individual components' center of gravity. However, in this study, these quantities were fixed at the baseline values. The object-oriented

	C_L, C_{D_i}, C_m	C_l, C_n, C_y	C_{D_f}
X_w	0.3–0.55	0.3–0.55	0.3–0.55
C_{r_w}(ft)	18.0–25.0	18.0–25.0	18.0–25.0
C_{t_w}(ft)	3.0–7.5	3.0–7.5	3.0–7.5
b_w(ft)	100.0–140.0	100.0–140.0	100.0–140.0
Λ_w(deg)	28.0–40.0	28.0–40.0	28.0–40.0
X_H	0.65–0.82	0.65–0.82	0.65–0.82
C_{r_H}(ft)	9.0–15.0	9.0–15.0	9.0–15.0
C_{t_H}(ft)	1.5–5.0	1.5–5.0	1.5–5.0
b_H(ft)	38.0–50.0	38.0–50.0	38.0–50.0
Λ_H(deg)	35.0–40.0	35.0–40.0	35.0–40.0
X_V	not used	not used	0.7–0.8
C_{r_V}(ft)	not used	18.0–21.0	18.0–21.0
b_V(ft)	not used	16.0–18.0	16.0–18.0
M	0.0–0.9	0.0–0.9	0.0–0.9
h(ft)	not used	not used	0.0-30k
α(deg)	-15.0–15.0	-15.0–15.0	not used
β(deg)	not used	-5.0–5.0	not used
δ_f(deg)	0.0–40.0	0.0–40.0	not used
δ_e(deg)	-20.0–20.0	-20.0–20.0	not used
δ_a(deg)	not used	-20.0–20.0	not used
δ_r(deg)	not used	-30.0–30.0	not-used

Table 1 Variables and ranges used for the aerodynamic metamodel

preprocessor developed for HASC and BDAP was extended to create FLOPS input files and 1500 random cases varying all geometric variables in Table 1 were obtained.

Neural Networks

References 24 and 25 demonstrate that Neural Networks behave as universal function approximators. In that respect, neural networks find uses and applications in all fields. Ross et al. have shown that they can be used to reduce wind tunnel data requirements by modeling the aerodynamic coefficients during wind tunnel tests.[26] Neural Networks have also found their niche in aerospace control applications.[27]

The computational unit in a Neural Network is the perceptron. The perceptron is responsible for mapping inputs into outputs with a set of weights, biases and a transfer function. It is depicted graphically in Figure 4. These computational units can be linked together into a larger network, a Neural Network. The way in which the perceptrons are connected to each other is called the *architecture* or *topology* of the network.

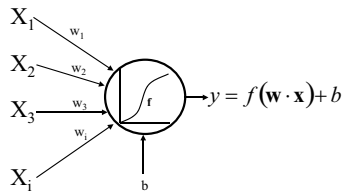


Fig. 4 Perceptron model

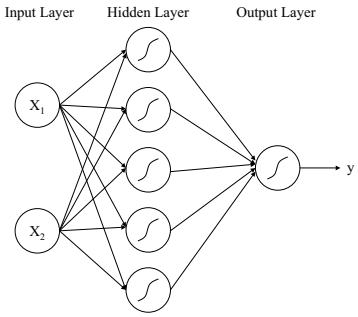


Fig. 5 Typical Neural Network Architecture

The heart of the Neural Network is the training procedure, an optimization algorithm responsible for minimizing the errors between the given and calculated outputs by varying the weights and biases. Given that the weights and biases are computed through training to guarantee the best prediction capability, the choice of architecture becomes the main degree of freedom in determining the best neural network in terms of prediction capability.

A typical neural network architecture is shown in Figure 5. Neural Networks typically have three or more layers: an input layer, one or more hidden layer and an output layer. The input layer is where the input signals are fed in and does not play a part in the design of the network. Also, there is one perceptron in the output layer for each output variable. Therefore, the topology is varied through the number of perceptrons in the hidden layer as well as the number of hidden layers. The number of degrees of freedom of a network is the sum of all weights and biases in the network.

Generalization

The ability of a Neural Network to predict responses accurately outside its training region is called *generalization*. It is a function of the number of DOF, and therefore its topology. However, the relationship is not linear. As the number of degrees of freedom increases, Neural Networks first go through a period of underfitting and can quickly reach a stage of overfitting. This

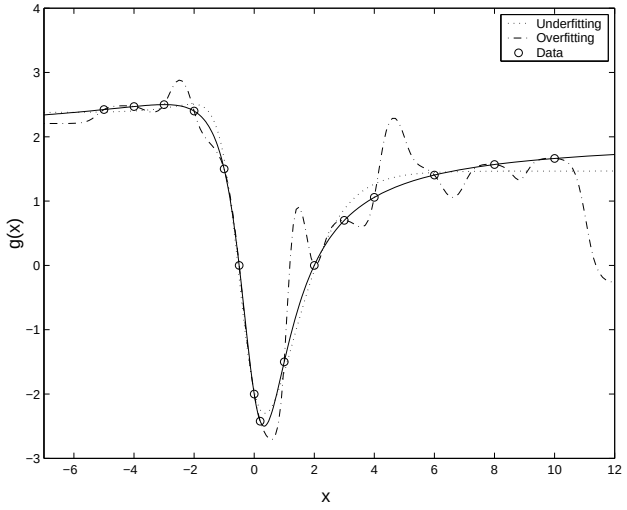


Fig. 6 Fitting power of NN for $g(x) = \frac{(x-2)(2x+1)}{(1+x^2)}$

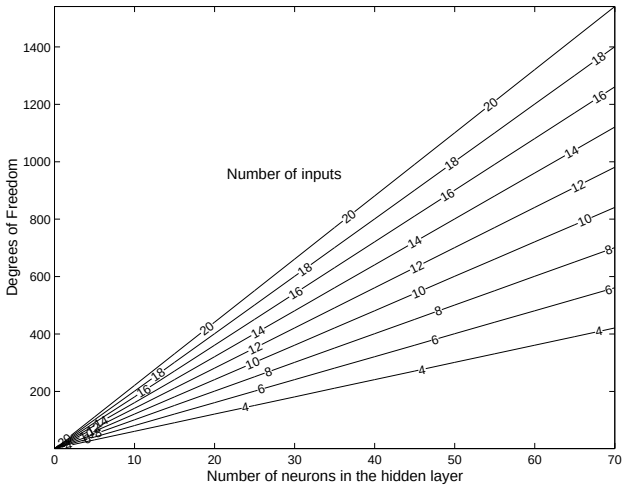


Fig. 7 Degrees of freedom of a single hidden layer feed-forward backpropagation network

can easily be depicted graphically for a function of one variable as in Figure 6. However, it becomes difficult to evaluate under- or over-fitting when there is a large number of input variables, such as may be the case for an aerodynamic model. In those cases, the input data can be split into two sets. The first set is used to train the neural network and the second set (usually much smaller) is used to validate the network by providing points that are not part of the training set. Prediction error comparison between the two sets can be used as a means to identify network generalization capability.

Generalization is also a function of the size of the training set. Hassoun has demonstrated that for good generalization, $m >> d$, where m is the size of the training set and d is the number of DOF.[28] The relationship between DOF, number of inputs and size of the hidden layer is shown in Figure 7. For example, for a network with 30 neurons and 14 inputs, good generalization is possible if $m >> 480$.

Two techniques have been widely used to improve generalization in neural networks. The first, called regularization, attempts to penalize large weights, which is a common source of poor generalization. This penalizing is done with an additional term in the training objective function. Instead of only minimizing training error, the optimization is done on a composite function (usually a weighted sum) that includes the error term and the sum of squares of the weights. However, a new free parameter needs to be introduced. The ratio of the error term weight to the weight of the penalization function is called the performance ratio or regularization parameter. Although the use of this new function as an objective during training improves generalization, it is difficult to estimate the optimum value for the new free parameter and is therefore tricky to use.[29] A technique for automatically setting the regularization parameter using a Bayesian framework has been implemented. However, its implementation is based on the Levenberg-Marquardt (LM) training algorithm, which is very fast and efficient for small to medium size networks but becomes very inefficient for large problems.[29] This is due to the fact that it needs to compute the approximate Hessian matrix and keep it in memory for each iteration.

Another empirically based technique for improving generalization is "Early Stopping". With this technique, the validation set is used during training to monitor the error on the validation set as training occurs. If the validation error increases for a fixed number of epochs in a row, training is stopped. This technique works very well for noisy data. It has been shown that early during training, neurons capture the general trends in the data. As training is prolonged, some neurons start to fit the noise and thereby compromise generalization.[28]

Sensitivity to initial conditions

Because the training process is based on optimization, it is sensitive to initial conditions. Furthermore, unless a global search optimizer is used, the optimum found is not guaranteed to be the global one. For these reasons it is necessary to train a network several times, starting at random initial weights and biases to hope for a global minimum and therefore the best possible network.

Selecting network architecture

This research used the *Matlab* software with the Neural Network Toolbox[29] for all its neural network applications. The neural network topology chosen was that of a single hidden layer, feed-forward backpropagation network with tangent-sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer. This topology is hypothetically capable of exactly mapping outputs to inputs given an infinite number of neurons in the hidden layer.

For a given training algorithm and dataset, network

prediction error is a pure function of the number of neurons in the hidden layer (N_h). If possible, for a given N_h, the network will train until a given training error goal is reached. Alternatively, the data can be presented a fixed number of times to the network (the number of epochs, N_e) with an asymptotic error goal of 0. Determining N_h for minimum error in the *validation* set becomes an integer based optimization problem in one dimension with increasingly expensive function calls corresponding to:

1. training the network up to a given goal (or to the fixed N_e),
2. using the network to obtain prediction error in the validation set,
3. repeat a fixed number of times (N_a) and average error to account for sensitivity to initial conditions.

This approach was used in determining N_h for the metamodels that will be used by the dynamic model. Regularization and early-stopping were not used in this study. Regularization introduces an additional free parameter which complicates the optimization further. Additionally, no noise is present in the collected data since it comes from computer simulations and is therefore 100% repeatable. Thus it is beneficial to let the networks train fully rather than stopping early.

Aerodynamics

The aerodynamic metamodel is comprised of seven networks, one for each aerodynamic coefficient in Table 1. The training set is chosen to contain the first 3000 random cases and the validation set contains the last 400 cases obtained from the HASC/BDAP aerodynamic model. This should allow for good generalization up to $N_h = 30$ using Hassoun's limit and Figure 7.

To determine the optimal N_h of each network, an improved grid search approach was developed. As N_h increases, the error in the training set and in the validation set are expected to decrease up to a point where overfitting starts to occur and the validation error increases. With that in mind, the search algorithm increases N_h within a given range at a fixed step. Training at each N_h is done N_a times to a fixed N_e and the validation error is averaged. N_h is increased until the mean absolute error in the validation step ceases to decrease. When this happens, the last three points are used to fit a quadratic curve whose minimum corresponds to N_h for minimum error in the validation set.

Once the optimum N_h has been determined, the corresponding network was retrained 10 times with random initial conditions. This ensured that only the best performing network, corresponding to the global minimum on the weight/bias error surface, was retained.

Propulsion

The engine deck generated by FLOPS for the baseline vehicle was used to train a neural network to model vehicle thrust as a function of Mach number, altitude and throttle setting. This problem is substantially smaller than its aerodynamic counterpart; it only has three independent variables and one output. Also, since all the data provided by FLOPS correspond to engine operating points, no validation data are available.

Due to the lesser complexity of this problem, the LM training algorithm with Bayesian regularization was used. This algorithm uses statistical techniques treating weights and biases as random variables around the LM optimization. By varying only those weights and biases that contribute to improving the error, the process guarantees a network with best generalization for a given N_h. Essentially, when using this algorithm, one is shielded from overfitting and therefore should aim for high N_h. The training error remains small with any value of N_h as it mainly affects the generalization capability of the network.

Mass properties

The complexity of the metamodeling problem for vehicle mass properties lies somewhere between those for the aerodynamics and propulsion problem. Although it has a number of independent variables roughly equal to that of the aerodynamic problem, its dependency is in the design variables alone and is therefore better behaved. The optimal N_h is expected to be smaller than that of the aerodynamic networks. The training set is comprised of the first 1200 random cases and the validation set contains the last 250 random cases generated from FLOPS.

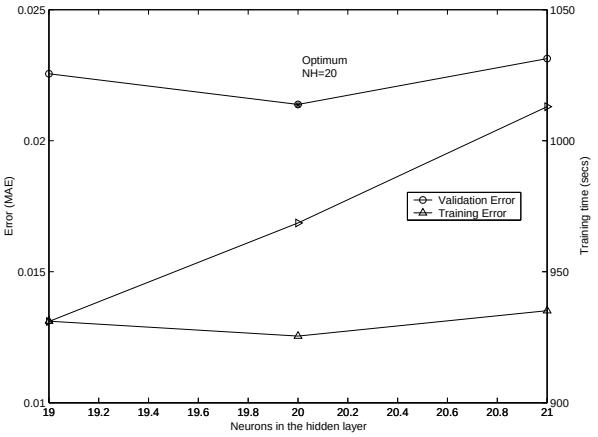
The same algorithm as for the aerodynamic networks was used to determine the optimum N_h for empty weight, and fuel weight. Again, after the optimum architecture is determined, each network was trained an additional 10 times, each time with random initial weights and biases, to retain the network with the least error in the validation set.

Results

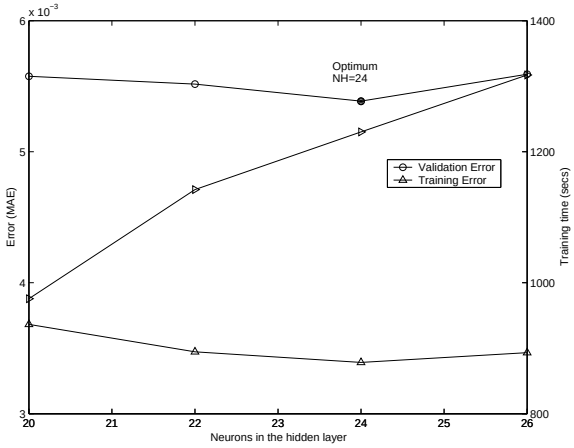
Aerodynamic metamodel

The results of the search for optimum N_h for the longitudinal aerodynamic coefficients are shown in Figure 8. This search was conducted using a scaled conjugate gradient training algorithm to a fixed $N_e = 3000$ with $N_a = 3$. The error performance of the optimized networks is shown in Figure 9 where they are also compared to second order RSEs modeled on the same datasets. Figure 9 shows that the neural networks perform on average three times better than the RSEs.

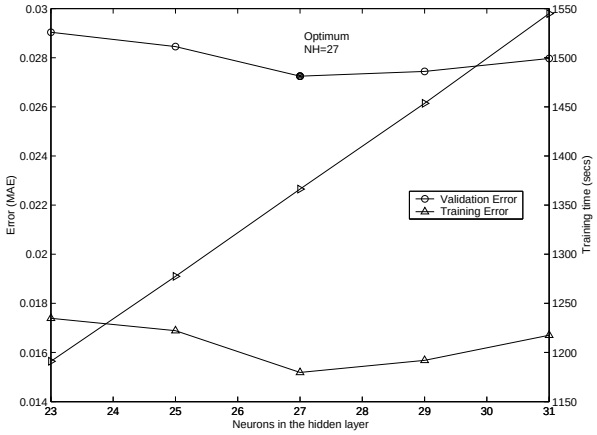
The complete parametric and probabilistic model for each aerodynamic coefficient included in the dy-



a) N_h **search for** C_L **network**



b) N_h **search for** C_{D_i} **network**



c) N_h **search for** C_m **network**

Fig. 8 N_h **optimization, longitudinal aerodynamics**

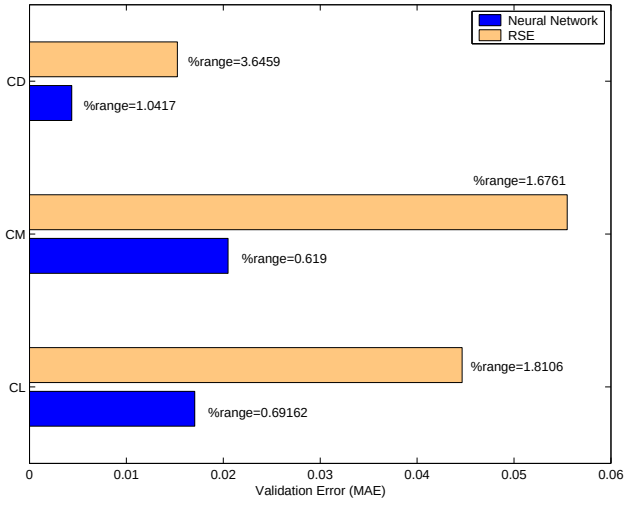


Fig. 9 Validation error performance of Longitudinal Aerodynamic metamodel

namic model is of the form:

$$C_x = C_{x_{metamodel}} + \epsilon_{model} + \epsilon_{metamodel} \qquad (3)$$

Where $\epsilon_{metamodel}$ is a random variable whose distribution is estimated from the known metamodel prediction performance. For example, the error statistics of the neural network for C_m is shown in Figure 10. The Figure shows error distribution for the validation set. The right hand side of the figure shows a normal distribution fit to the error data. Although not exactly normal, the error distribution fits very closely to a normal distribution except at the tails.

Various techniques have been used to estimate ϵ_{model}[30,31] . These techniques can be used alongside known model validation data[21,22] to estimate the characteristics of the probability density function making up the random variable ϵ_{model}. This was not done for this study.

Propulsion metamodel

The network for thrust with $N_h = 35$ is shown in Figure 11. The lines correspond to network output and

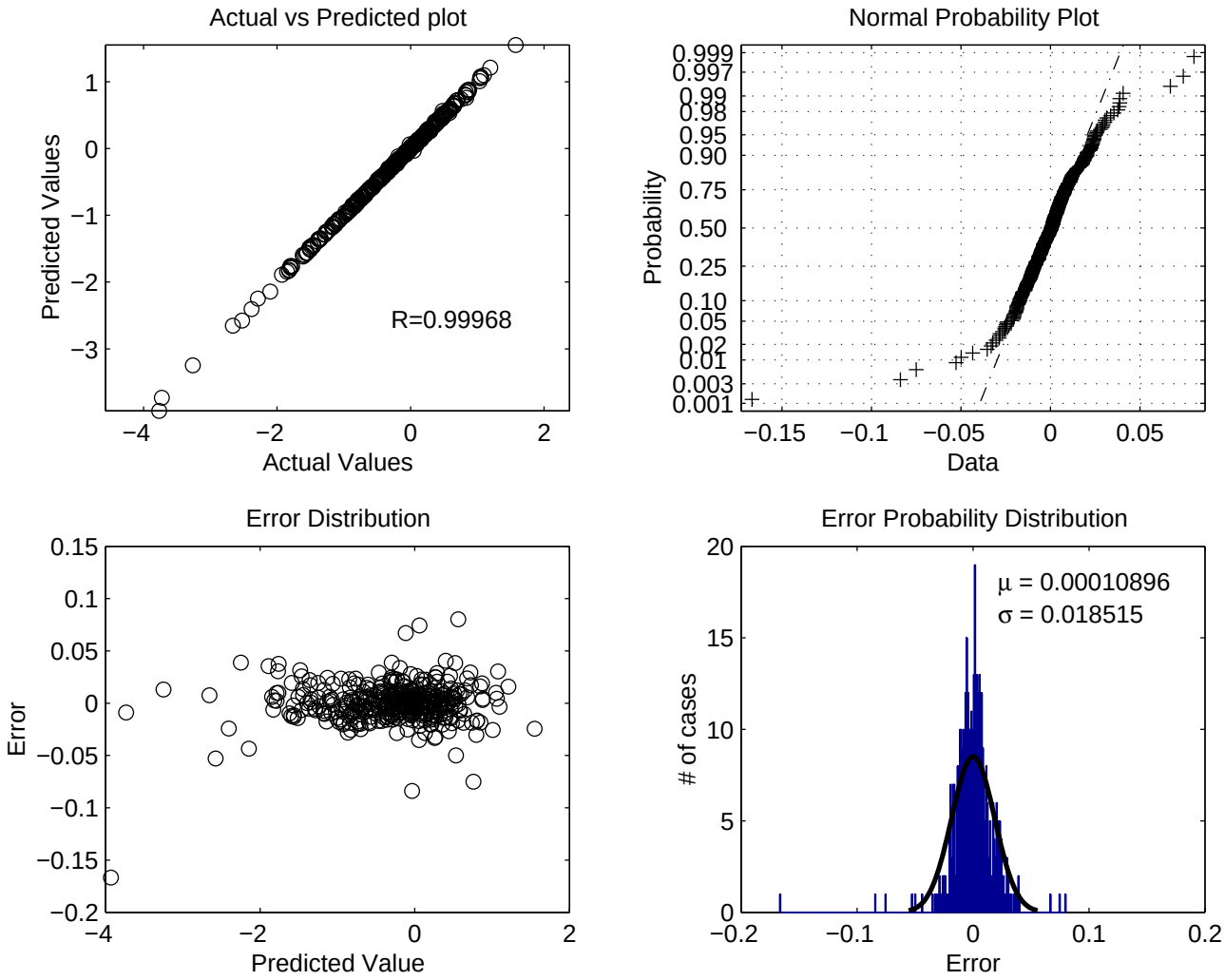
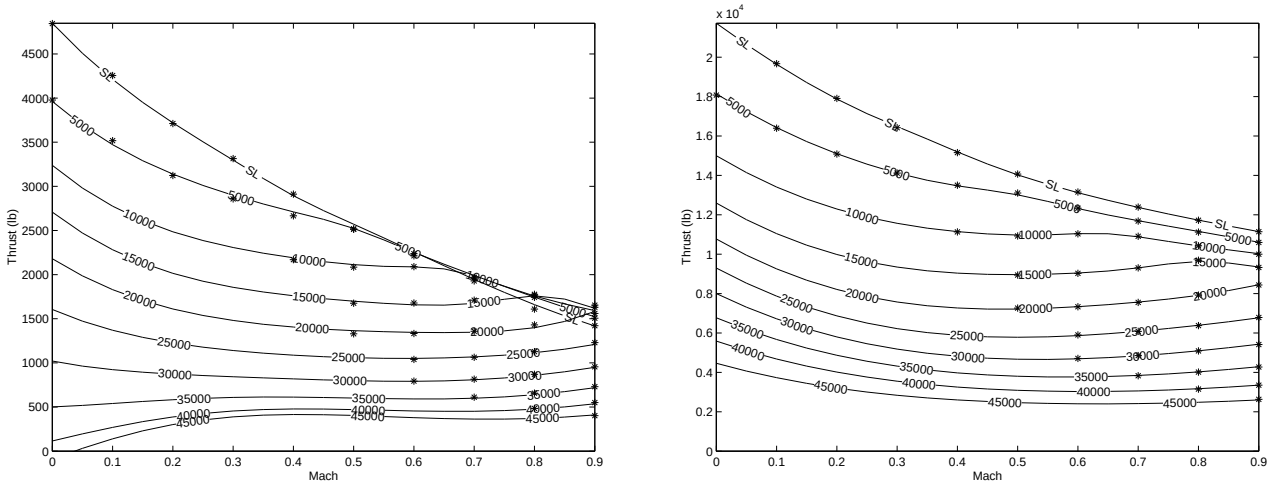
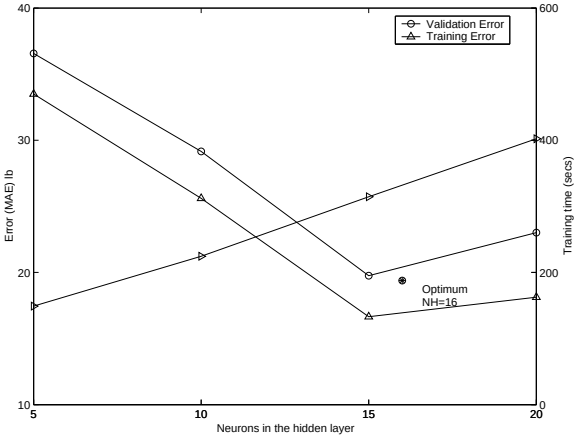


Fig. 10 Error statistics of the C_m neural network

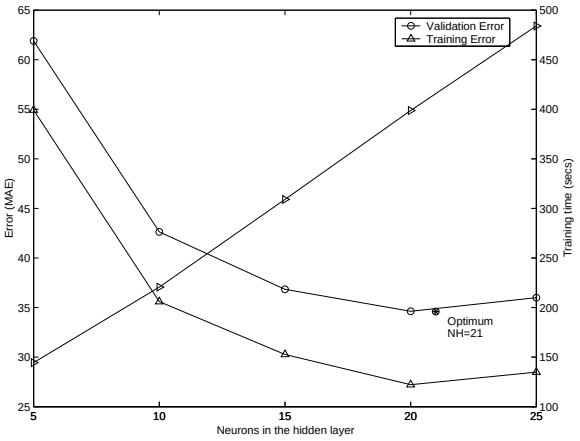


a) Throttles at 20% **b) Full Throttle**

Fig. 11 Neural Network model for thrust as function of altitude and Mach number



a) Empty Weight



b) Fuel Weight

Fig. 12 N_h optimization, mass properties

the stars correspond to the input dataset. The mean absolute error (on the validation set) of the network is $23lb$ or 0.1% of the entire thrust range. Figure 11 shows very good interpolation. The input dataset correspond to points where the engine can be operated and therefore the extrapolation region is of little interest since the engine is not made to operate there. The complete parametric and probabilistic model for thrust is of the form of Equation 3 and is generated as discussed above.

Mass properties metamodel

The optimization on N_h for empty weight and fuel weight is shown in Figure 12. As was the case for the aerodynamic networks, the search was conducted using a scaled conjugate gradient training algorithm to a fixed $N_e = 3000$ with $N_a = 3$. The error performance of the optimized networks is shown in Figure 13. Again, the complete parametric and probabilistic model for the mass properties is of the form of Equation 3.

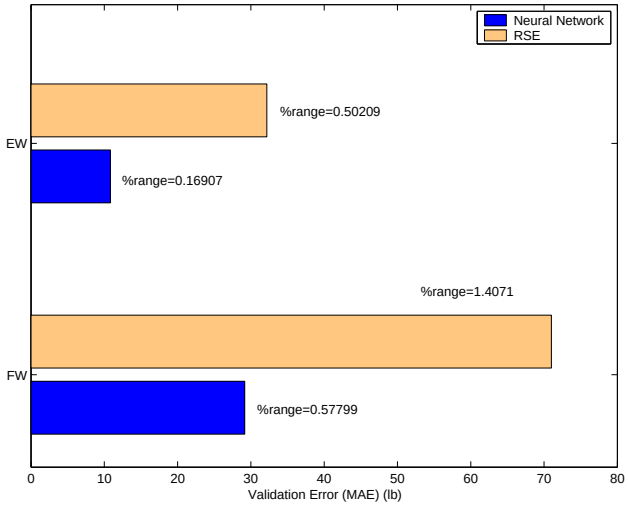


Fig. 13 Validation errorPerformance of mass properties metamodel

Conclusions

Metamodels to be included within a parametric dynamic vehicle model were created using neural networks. Aerodynamics, propulsion and mass properties models were used to generate training data for those neural networks. The central part in obtaining networks that generalize well was the search for the optimal network architecture in terms of the number of neurons in the hidden layer. An incremental grid search algorithm was developed to determine the optimum architecture.

The resulting neural networks performed far better than second order Response Surfaces. However, better generalization comes at the expense of model building time. For $N_h = 20$ it took on average 15 minutes to train a network with 3000 data points up to 3000 epochs. By comparison, a second order response surface can be generated in less than a minute for the same dataset. Yet, for problems of large or unknown complexity such as the aerodynamics problem in this study, Response Surfaces are not powerful enough metamodels to warrant their use within a parametric dynamic vehicle model.

Further research is needed to find ways to reduce the time needed to arrive at the dynamic vehicle model. Both improved data collection (sampling) and input/output transformations are means to improve both metamodel fit and training time. Also, further research in neural network architecture such as the use of Radial Basis networks or additional hidden layers in back-propagation networks, may lead to improved training times. Lastly, correlations between model parameter sensitivity and model fidelity need to be explored. For example, dynamic output is expected to be very sensitive to changes in inertia properties as well as center of gravity positions. Yet, these quantities carry the most uncertainty in early phases of the design process. Probabilistic design techniques can be

used to quantify the probability of a dynamic output given the level of uncertainty and fidelity in the design data.

A complete parametric and probabilistic vehicle dynamic model can now be obtained using neural networks in less than 3 full days on a 1Ghz personal computer. This model represents full vehicle dynamics for the entire design space. The dynamic model can be used in design studies to evaluate vehicle dynamics and control parametrically and probabilistically, a much needed improvement over traditional dynamic analyses. The authors believe that the value of such a model for use during conceptual design far outweighs its cost.

References

[1]Mavris, D. N., DeLaurentis, D. A., Bandte, O., and Hale, M. A., "A stochastic Approach to multi-disciplinary aircraft analysis and design," AIAA, 1998.

[2]Mavris, D. N. and DeLaurentis, D. A., "A stochastic approach for aicraft affordability," ICAS, 1998.

[3]DeLaurentis, D., Mavris, D. N., and Schrage, D. P., "System synthesis in preliminary aircraft design using statistical methods," ICAS, 1996.

[4]Mavris, D. N., DeLaurentis, D. A., Hale, M. A., and Tai, J. C., "Elements of an emerging virtual stochastic life cycle design environment," No. 1999-01-5638, AIAA/SAE, 1999.

[5]AGARD conf. proceedings, *Flight Simulation — Where are the Challenges?*, No. AGARD-CP-577, Braunschweig, Germany, 1995.

[6]"Intelligent Synthesis Environment, Program Plan," On the web at http://www.ise.nasa.gov, September 2000, Version 1.0.

[7]Johnson, M. V., McKeon, M. F., and Szanto, T. R., "Simulation based acquisition: A new Approach," Tech. rep., Defense Systems Management College, December 1998.

[8]Mavris, D. N., DeLaurentis, D. A., and Soban, D. S., "Probabilistic assessment of handling qualities characteristics in preliminary aircraft design," *36th Aerospace Sciences Meeting & Exhibit*, No. AIAA 98-0492.

[9]DeLaurentis, D. A., Mavris, D. N., Calise, A. J., and Schrage, D. P., "Generating dynamic models including uncertainty for use in aircraft conceptual design," *AIAA Atmospheric Flight Mechanics Conference*, No. AIAA 97-3590.

[10]Scharl, J., Mavris, D., and Burdun, I. Y., "Use of Flight Simulation in Early Design: Formulation and Application of the Virtual Testing and Evaluation Methodology," No. 2000-01-5590, SAE, 2000.

[11]Burdun, I. Y., "An AI situational pilot model for real-time applications," *20th Congress of the International Council of the Aeronautical Sciences*, Vol. I, Sorrento, Italy, 8-13 September 1996, pp. 210–237.

[12]Box, G. E. and Draper, N. R., *Empirical model-building and response surfaces*, John Wiley & Sons, 1987.

[13]Mavris, D. N., Bandte, O., and Schrage, D. P., "Application of probabilistic methods for the determination of an economically robust HSCT configuration," AIAA, 1996.

[14]Weiss, S. and Thielecke, F., "Aerodynamic model identification using local model networks," No. 2000-4098, AIAA, 2000, pp. 364–372.

[15]Daberkow, D. D. and Mavris, D. N., "New Approaches to conceptual and preliminary aircraft design: a comparative assessment of a neural network formulation and a response surface methodology," No. 985509, AIAA/SAE, September 1998.

[16]Papila, N. et al., "Assessment of neural net and polynomial-based techniques for aerodynamic applications," No. 99-3167, AIAA, 1999.

[17]Vaidyanathan, R., Papila, N., Shyy, W., Tucker, P. K., and Griffin, L. W., "Neural Network and response surface methodology for rocket engine component optimization," No. 2000-4880, AIAA, 2000.

[18]Etkin, B. and Reid, L. D., *Dynamics of Flight*, Wiley, 1995.

[19]Nelson, R. C., *Flight Stability and Automatic Control*, McGraw-Hill, 1990.

[20]Rolfe, J. M. and Staples, K. J., *Flight Simulation*, Cambridge Aerospace Series, Cambridge University Press, 1986.

[21]Albright, A. E., Dixon, C. J., and Hegedus, M. C., "Modification and Validation of Conceptual Design Aerodynamic Prediction Method HASC95 With VTXCHN," Tech. rep., NASA Contract Report 4712, 1996.

[22]Boeing, *Boeing Drag Analysis Program, BDAP*.

[23]McCullers, A., *Flight OPtimization System*, NASA Langley, version 5.94.

[24]Hagan, M. T., Demuth, H. B., and Beale, M., *Neural Network Design*, PWS Publishing, 1996.

[25]Prinicipe, J. C., Euliano, N. R., and Lefebvre, C. W., *Neural and Adaptive Systems*, Wiley & Sons, 1999.

[26]Ross, J. C., Jorgenson, C. C., and Nørgaard, M., "Reducing Wind Tunnel Data Requirements Using Neural Networks," Tech. rep., NASA TM-112193, May 1997.

[27]Vim, B. S. and Calise, A. J., "Nonlinear flight control using neural networks," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, Jan.-Feb. 1997, pp. 26–33.

[28]Hassoun, M. H., *Fundamentals of Artificial Neural Networks*, MIT Press, 1995.

[29]Demuth, H. and Beale, M., *Neural Network Toolbox for use with Matlab*, The Math Works, Inc.

[30]Hayden, W.T., Mavris, D.N., "Probabilistic Assessment of the Impact of New Technologies on an HSCT," No. AIAA-1999-01-5633 in First Internet Conference on Approximations and Fast Reanalysis in Engineering Optimization, The International Society for Structural and Multidisciplinary Optimization (ISSMO), the NASA Langley Research Center, and the American Institute of Aeronautics and Astronautics (AIAA), Conducted entirely on the Internet, June 1998.

[31]DeLaurentis, D. A., *A Probabilistic Approach to Aircraft Design Emphasizing Stability and Control Uncertainties*, Ph.D. thesis, Georgia Institute of Technology, 1998.